Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №4

Технології розроблення програмного забезпечення

Тема: Вступ до паттернів проектування

Виконав
студент групи ІА-34:
Жительний В.В.

Перевірив:
Мягкий М.Ю.

Київ 2025

Мета: Вивчити структуру шаблонів «Singleton», «Iterator», «Proxy», «State», «Strategy» та навчитися застосовувати їх в реалізації програмної системи.

Тема:

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

Хід роботи

Для моєї теми «Онлайн радіо станці» було вибрано реалізувати шаблон «Ітератор», оскільки в реалізації присутні плейлисти та треки, і задля полегшення ітерування по ним, і було вибрано цей шаблон. Завдяки цьому можна буде реалізувати різні способи перебору колекцій та багато іншого.

На наступній діаграмі класів продемонстровано структуру даного шаблону для моєї теми:

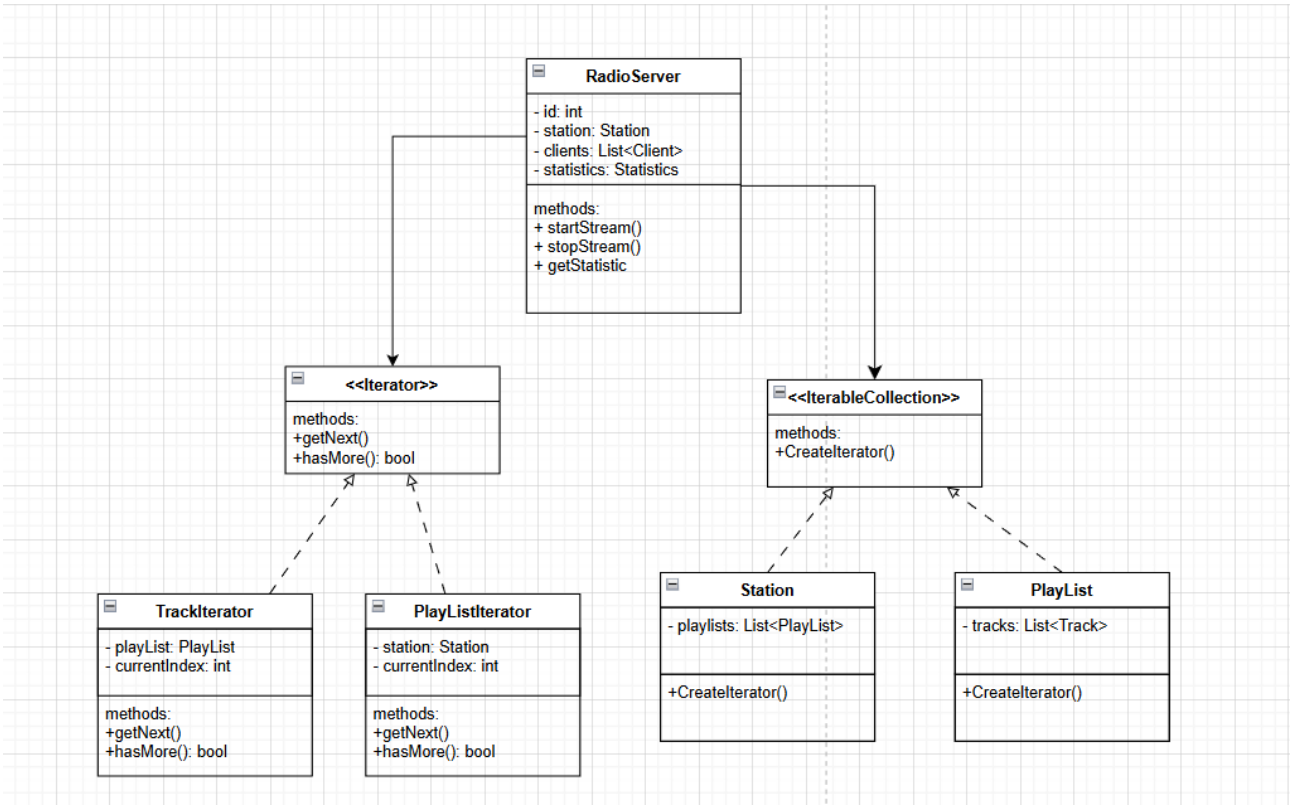


Рис. 1 Діаграма класів для шаблону Ітератор

Коротко описуючи діаграму, в нас присутній основний клас, або ж клієнт, який працює з усіма об’єктами через інтерфейси колекції та ітератора. Також присутні два інтерфейси – ітератор та ітераційна колекція. Ітератор описує інтерфейс для доступу та ітерування елементів в колекції, а Ітераційна колекція описує інтерфейс створення ітератора. Останні елементи діаграми це реалізації вищезгаданих інтерфейсів, які реалізують їхні методи.

Програмна реалізація

Інтерфейс Ітераційна колекція:

```
public interface IterableCollection {
    Iterator<T> createIterator();
}
```

Інтерфейс Ітератор:

```
public interface Iterator {
    boolean hasNext();

    T next();
}
```

Реалізація для класу Playlist для ітерації по треках

Клас Playlist реалізує IterableCollection, що означає, що він надає ітератор для проходження по об'єктах типу Track.

Реалізація патерну в класі Playlist використовує внутрішній клас PlaylistTrackIterator. Такий підхід обрано з метою інкапсуляції: логіка ітераціїповністю прихована від зовнішнього коду. Playlist надає лише публічний метод createIterator(), який повертає об'єкт ітератора, що відповідає загальному інтерфейсу

```
public class Playlist implements IterableCollection<Track> {
    private Long id;
    private String name;
    private List<Track> tracks = new ArrayList<>();

    public List<Track> getTracks() {
        return tracks;
    }

    @Override
    public Iterator<Track> createIterator() {
        return new PlaylistTrackIterator(this);
    }

    private class PlaylistTrackIterator implements Iterator<Track> {
        private List<Track> tracksToIterate;
        private int currentIndex = 0;


        public PlaylistTrackIterator(Playlist playlist) {
            this.tracksToIterate = new ArrayList<>(playlist.getTracks());
```

```
        }


        @Override
        public boolean hasNext() {
            return !tracksToIterate.isEmpty();
        }


        @Override
        public Track next() {
            if (!hasNext()) {
                throw new NoSuchElementException("Плейлист порожній");
            }

            if (currentIndex >= tracksToIterate.size()) {
                currentIndex = 0;
            }

            Track track = tracksToIterate.get(currentIndex);
            currentIndex++;
            return track;
        }
    }
}
```

Реалізація для класу Station для ітерації по плейлистах

Клас Station реалізує IterableCollection, що означає, що він надає ітератор для проходження по об'єктах типу Playlist:

```
public class Station implements IterableCollection<Playlist> {
    private Long id;
    private String name;
    private List<Playlist> playlists = new ArrayList<>();

    public List<Playlist> getPlaylists() {
        return playlists;
    }


    @Override
    public Iterator<Playlist> createIterator() {
        return new StationPlaylistIterator(this);
    }


    private class StationPlaylistIterator implements Iterator<Playlist> {
        private List<Playlist> playlistsToIterate;
        private int currentIndex = 0;

        public StationPlaylistIterator(Station station) {
            this.playlistsToIterate = new ArrayList<>(station.getPlaylists());
        }


        @Override
        public boolean hasNext() {
            return !playlistsToIterate.isEmpty();
        }
```

```
        @Override
        public Playlist next() {
            if (!hasNext()) {
                throw new NoSuchElementException("Station has no playlists");
            }

            if (currentIndex >= playlistsToIterate.size()) {
                currentIndex = 0;
            }

            Playlist playlist = playlistsToIterate.get(currentIndex);
            currentIndex++;
            return playlist;
        }

}
```

Контрольні питання:

1. Що таке шаблон проєктування?

Шаблон проєктування – це, так би мовити, ескіз перевірений часом, за якими можна зручно проєктувати у відповідних обставинах.

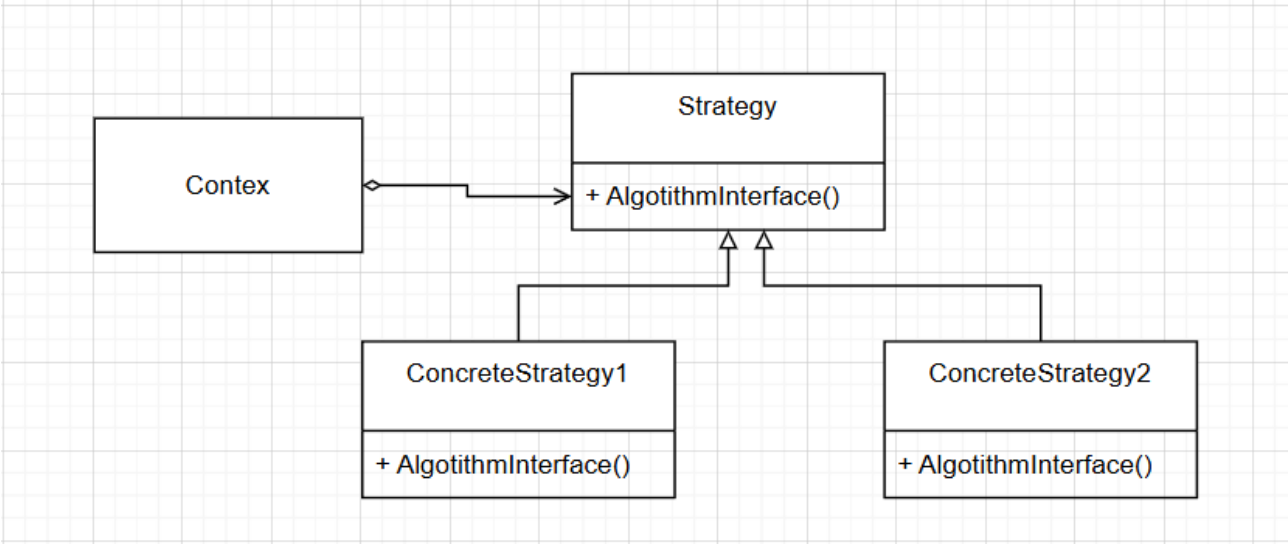
2. Навіщо використовувати шаблони проєктування?

Застосування шаблонів проєктування набагато спрощує впровадження нових змін до системи, а також враховуючи що шаблони досить уніфіковані, через що, можна дуже легко розуміти що треба робити, коли система проєктується в команді.

3. Яке призначення шаблону «Стратегія»?

Основне призначення цього алгоритму додати можливість об’єкту змінювати алгоритм поведінки іншим алгоритмом. Наприклад для використання різних алгоритмів сортування.

4. Нарисуйте структуру шаблону «Стратегія».



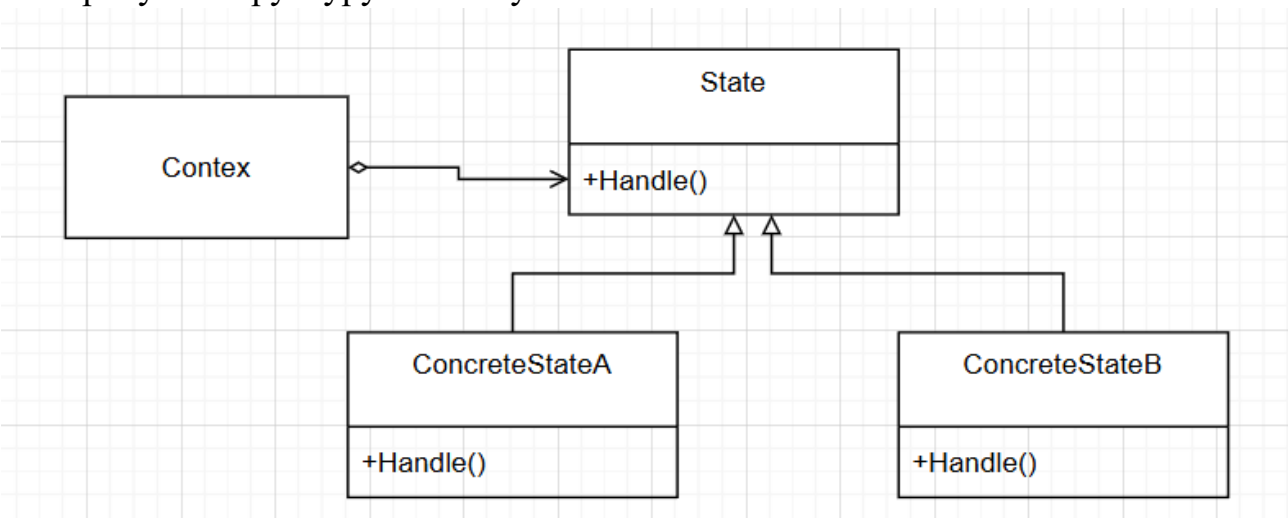
5. Які класи входять в шаблон «Стратегія», та яка між ними взаємодія?

До цього шаблону входять два основних класи Contex та Strategy. Contex зберігає посилання на об’єкт Strategy. Коли клієнт викликає метод у Contex, Contex викликає метод на своєму об’єкті Strategy.

6. Яке призначення шаблону «Стан»?

Основне призначення цього шаблону – це дати можливість об'єкту змінювати свою поведінку, коли змінюється його внутрішній стан

7. Нарисуйте структуру шаблону «Стан».



8. Які класи входять в шаблон «Стан», та яка між ними взаємодія?

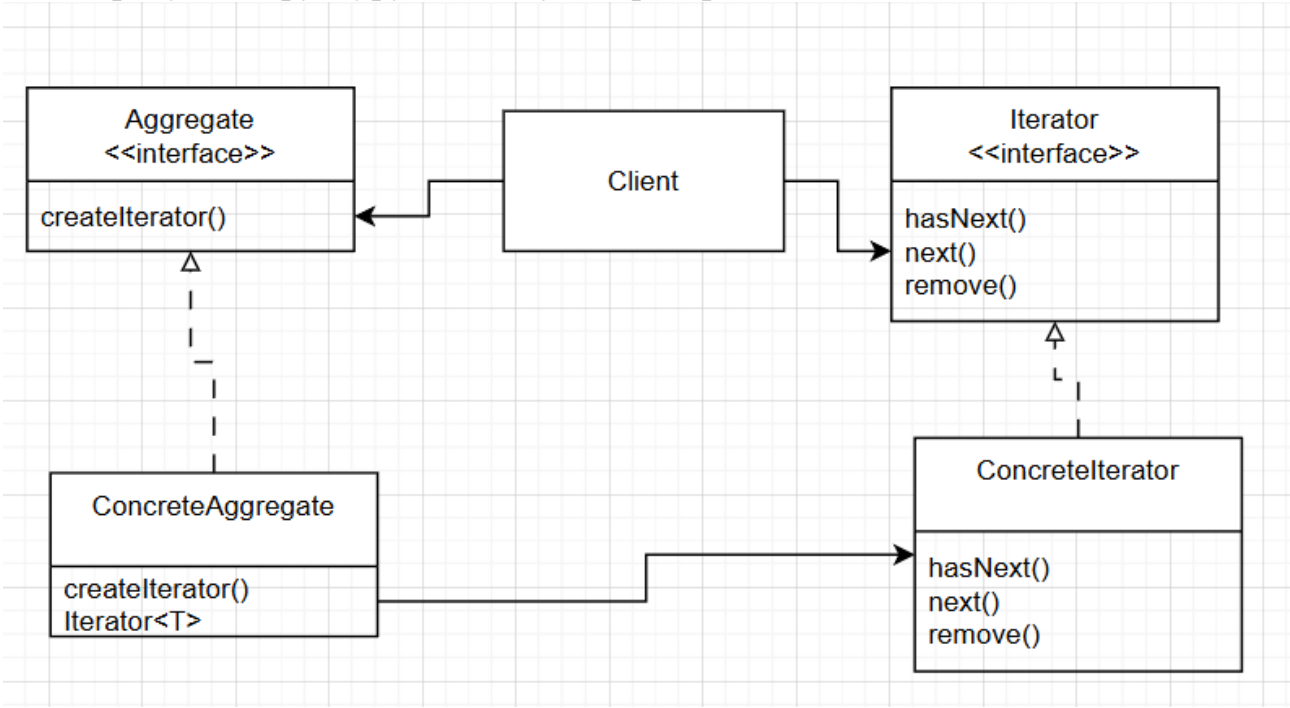
В цей шаблон входять класи Contex та State. Contex делегує всі запити, що залежать від стану, поточному об’єкту State. А об’єкт State виконує логіку і

може змінити поточний стан Contex, наприкалад підставивши замість себе новий об’єкт стану.

9. Яке призначення шаблону «Ітератор»?

Основне призначення шаблону ітератор, це винести логіку ітерування з самої колекції для розділення обов’язків, тобто колекція відповідає за зберігання елементів, а ітератор за перебирання.

10. Нарисуйте структуру шаблону «Ітератор».



11. Які класи входять в шаблон «Ітератор», та яка між ними взаємодія?

До шаблону ітератор входять два основні класи – сама колекція по якій можна ітеруватися, та сам ітератор, який по ній ітерується.

12. В чому полягає ідея шаблона «Одинак»?

Основна ідея шаблону «Одинак» - це те, що існує один об’єкт класу, і не можливо створити інший об’єкт, при спробі створити новий об’єкт даного класу, просто повернеться існуючий. Даний шаблон часто застосовується, коли необхідно мати лише конкретну кількість об’єктів або коли треба жорстко контролювати всі операції, що проходять через даний клас.

13. Чому шаблон «Одинак» вважають «анти-шаблоном»?

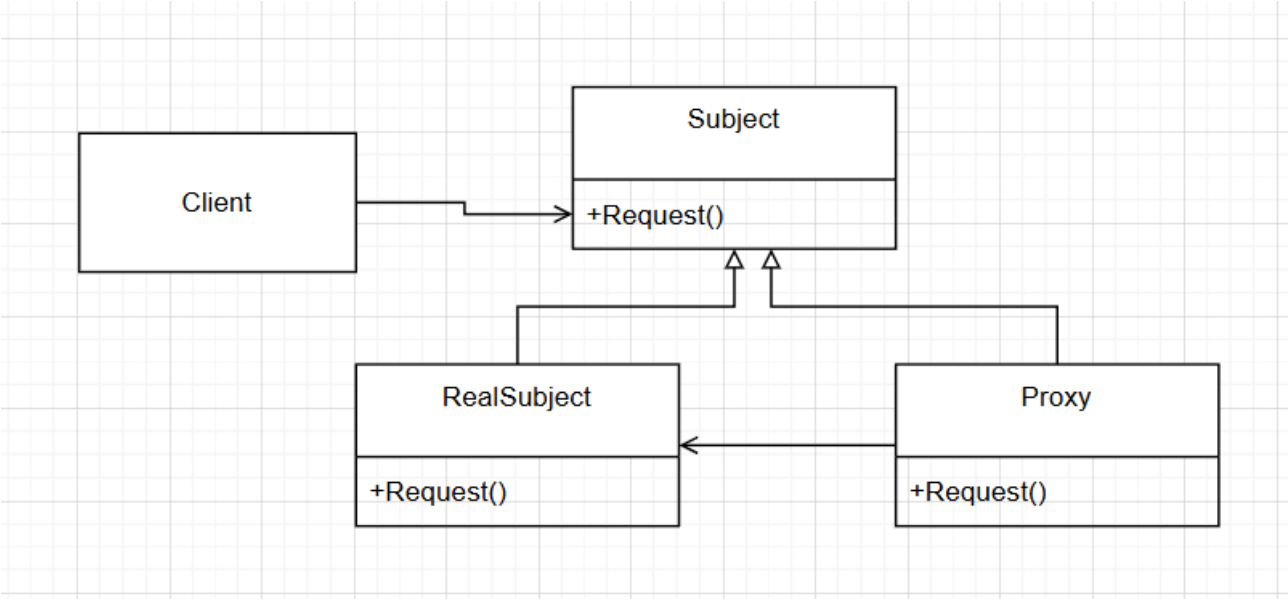
Цей шаблон вважають «анти-шаблоном», оскільки об’єкти з використанням цього шаблону працюють як глобальні змінні, через що, їх дуже важко

відслідковувати та змінювати, оскільки кожна зміна призводить до зміни купи коду.

14. Яке призначення шаблону «Проксі»?

Цей шаблон вирішує проблему, коли треба додати якусь логіку до або після виклику методів якогось об’єкта, не змінюючи код самого об’єкта

15. Нарисуйте структуру шаблону «Проксі».



16. Які класи входять в шаблон «Проксі», та яка між ними взаємодія?

Основні класи в цьому шаблоні це Subject, RealSubject та Proxy. Клієнт працює з об’єктами через інтерфейс Subject, він думає, що працює з реальним об’єктом, але насправді взаємодіє з Proxy.

Висновок: Під час виконання даної лабораторної роботи було досягнуто поставлену мету: вивчено та проаналізовано групу шаблонів проєктування, що використовуються для вирішення типових задач в об'єктно-орієнтованому програмуванні. Було детально розглянуто призначення, структуру та принципи взаємодії класів для наступних патернів: «Стратегія», «Стан», «Ітератор», «Одинак» та «Проксі». А також на практиці в програмному коді було реалізовано шаблон «Ітератор».